REMARKS

In the Official Action mailed on **24 April 2009**, the Examiner reviewed claims 1-2, 4-11, 13-18, and 28-35. Examiner objected to claims 1-2, 4-11, 13-18, and 28-35 because of informalities. Examiner rejected claims 28-35 under 35 U.S.C. § 112. Examiner rejected claims 1-2, 4-7, 10-11, 13-16, and 28-35 under 35 U.S.C. § 103(a) based on Kwong (U.S. Patent No. 6,289,506, hereinafter "Kwong"), and Ghosh (U.S. Patent No. 6,412,109, hereinafter "Ghosh"). Examiner rejected claims 8 and 17 under 35 U.S.C. §103(a) based on Kwong, Ghosh, and Kilis (U.S. Patent No. 5,491,821, hereinafter "Kilis"). Examiner rejected claims 9, and 18 under 35 U.S.C. § 103(a) based on Kwong, Ghosh, and Evans et al. (U.S. Patent No. 5,805,899, hereinafter "Evans").

Claim Objections

Claims 1-2, 4-11, 13-18, and 28-35 are objected to because they include informalities. Specifically, Examiner avers that claims 1, 2, 4-6, 9-11, 13-15, 18, 28-29, 31-33 and 35 recite the limitation "the native code method," which lacks proper antecedent basis. Accordingly, Applicant has amended these claims as suggested by Examiner. No new matter has been added.

Examiner avers that claims 1, 10, 28, and 32 recite the limitation "the calls," which lacks proper antecedent basis. Accordingly, Applicant has amended claims 1 and 10 as suggested by Examiner. No new matter has been added. Furthermore, Applicant has amended claims 28 and 32 to replace the phrase "the calls" with the phrase "the calls **from the selected native code method to the application**," which corresponds to a callback. No new matter has been added.

Examiner avers that claims 2, 9, 11, and 18 recite the limitation "the call to the native code method," which lacks proper antecedent basis. Accordingly, Applicant has amended claims 2 and 11 as suggested by Examiner. Furthermore,

Applicant has amended claims 9 and 18 to replace the phrase "the call to the native code method" with the phrase "the call to the **selected** native code method." No new matter has been added.

Examiner avers that claims 6, 15, 31, and 35 recite the limitation "combining," which lacks proper antecedent basis. Accordingly, Applicant has amended these claims as suggested by Examiner. No new matter has been added.

Examiner avers that claims 28, 30, 32, and 34 recite the limitation "the callback by the native code method," which lacks proper antecedent basis. Accordingly, Applicant has amended these claims as suggested by Examiner. No new matter has been added.

Rejections under 35 U.S.C. § 112

Claims 28-35 are rejected under 35 U.S.C. § 112 for being indefinite. Specifically, Examiner avers that claims 28 and 32 recite the limitation "the interactions," which lacks proper antecedent basis. Accordingly, Applicant has amended claims 28 and 32 to replace the phrase "the interactions" with the phrase "the **callback**." Furthermore, Applicant has replaced the phrase "involves optimizing calls from the application to the native code method" with the correct phrase "involves optimizing calls **from the selected native code method to the application**" that corresponds to a callback. These amendments find support in paragraphs [0024] and [0031]-[0032] of the instant application. No new matter has been added.

Examiner also avers that claim 32 recites the limitation "the combined intermediate representation," which lacks proper antecedent basis. Accordingly, Applicant has amended claim 32 to replace the phrase "combined" with the phrase "**integrated**." This amendment finds support in paragraphs [0030]-[0031] and FIG. 2 of the instant application. No new matter has been added.

Rejections under 35 U.S.C. § 103(a)

Claims 1-2, 4-7, 10-11, 13-16, and 28-35 are rejected under 35 U.S.C. § 103(a) as being obvious in view of Kwong and Berry. In rejecting the independent claims, Examiner avers that Kwong discloses:

> "integrating the **intermediate representation** for the selected native code method into the intermediate representation associated with the application running on the virtual machine to form an **integrated intermediate representation** (see Figure 7: 705 and 740; Column 10: 8-10, '… the **Java application now comprises of Lib.dll 1060, A.class 1010, and B.class 1020** and may be **executed** on a Java VM 1080.");" (See Office Action dated 24 April 2009, emphasis added).

Applicant respectfully disagrees, because the cited sections of Kwong at most disclose **compiling an executable Java application** from a combination of Java code and native code, which is fundamentally distinct from generating an integrated intermediate representation.

Applicant avers that "intermediate representation" has a specific meaning within the field of software compilers. Specifically, one with common knowledge in the art will understand that an intermediate representation is generated by a software compiler as an initial step such that "linear human-readable text (i.e., source code) representing a program is transformed into an **intermediate graph data structure** that allows **flow analysis and re-arrangements *before* starting to create the list of actual CPU instructions** that will do the work" (see, e.g., "Intermediate representation" *Wikipedia, The Free Encyclopedia*. 3 Jul 2009, 21:23 UTC. 3 Jul 2009 <http://en.wikipedia.org/w/index.php?title=Intermediate_representation&oldid=300119373>.). In other words, an intermediate representation is **a data structure** which can be optimized during an intermediate compiler operation, which is performed **prior to generating the executable binary instructions** from the intermediate representation.

Applicant respectfully notes that Kwong does not disclose in any way generating an **integrated intermediate representation**. Kwong discloses a system which **compiles** a Java application with Dynamic Link Libraries **into an executable** Java application, and monitors the performance of program code **during program execution** (see Kwong, columns 9-10 and abstract). Kwong explicitly states at the bottom of Column 9 that A.java 1005 and B.java 1015 **"are compiled with a Java compiler 1030 to generate A.class 1010 and B.class 1020**, respectively…" which "**may then be executed** on a Java VM 1080" (see Kwong, Column 9, line 65 – Column 10, line 3). Furthermore, a Dynamic Link Library (e.g., Lib.dll) is commonly known in the art to be a pre-compiled library of **executable binary code**. Therefore, the cited section of Column 10 in Kwong at most discloses combining multiple sources **of executable binary code** into a **Java application**, which is fundamentally distinct from generating an **intermediate representation**.

Applicant further notes that the cited compiler operations illustrated in Figure 7 of Kwong in no way hint at generating an integrated intermediate representation from native binary code and from original source code. At most, Figure 7 in Kwong illustrates the operations described in columns 9 and 10 of Kwong, which explicitly describes compiling source code files **to generate executable files**, and then combining these compiled executables with Dynamic Link Libraries **to generate an executable Java application**.

Therefore, Kwong does not disclose in any way generating an **integrated intermediate representation**, because: 1) Kwong compiles Java source files **into executable objects** (e.g., A.class and B.class); 2) Kwong combines the compiled executable objects **with pre-compiled Dynamic Link Libraries** (e.g., Lib.dll) to generate an **executable** Java application; and 3) the **executable** Java application is in no way similar to an **intermediate representation**, because an intermediate representation is by definition **not in final executable form**.

In contrast, embodiments of the present invention provide a system for reducing the overhead involved in executing native code methods in an application that runs on a virtual machine. To do so, this system generates an intermediate representation for the application, and generates an intermediate representation for a selected native code method. Then, the two intermediate representations are integrated to form a single integrated intermediate representation, which is then used to optimize calls to and from an application that runs on a virtual machine to the native code method. Applicant respectfully notes that an intermediate representation, such as the integrated intermediate representation, includes a set of instruction code which is **not in final executable form** (see instant application, paragraph [0028] and FIG. 2).

Kwong and Berry nowhere describe, either explicitly or implicitly, generating an **integrated intermediate representation** from source code for an application that runs on a virtual machine, and from decompiled native code, such that the intermediate representation includes a set of instruction code which is **not in final executable form**.

Accordingly, Applicant has amended claims 1, 10, 28 and 32 to clarify that an intermediate representation includes a set of instruction code which is **not in final executable form**. These amendments find support in paragraph [0028] of the instant application. No new matter has been added.

Hence, Applicant respectfully submits that independent claims 1, 10, 28, and 32 are in condition for allowance. Applicant also submits that claims 2 and 4-9, which depend upon claim 1, claims 11 and 13-18, which depend on claim 10, claims 29-31, which depend on claim 28, and claims 33-35, which depend on claim 32, are for the same reasons in condition for allowance and for reasons of the unique combinations recited in such claims.

CONCLUSION

It is submitted that the present application is presently in form for allowance. Such action is respectfully requested.

Respectfully submitted,

By ___/Anthony Jones/_____
Anthony Jones
Registration No. 59,521

Date: 22 July 2009

Anthony Jones
Park, Vaughan & Fleming LLP
2820 Fifth Street
Davis, CA 95618-7759
Tel: (530) 759-1666
Fax: (530) 759-1665
Email: tony@parklegal.com